Extract Topics from Text Data

Topic modeling is a type of Natural Language Processing (NLP) task that utilizes unsupervised learning methods to extract out the main topics of some text data we deal with. Instead of pre-training on the data that have associated topic labels, the algorithms try to discover the underlying patterns, in this case, the topics, directly from the data itself.

Here are the requirements.

```
import nltk
from nltk.stem import *
nltk.download(punkt) # For Stemming
nltk.download(wordnet) # For Lemmatization
nltk.download(stopwords) # For Stopword Removal
nltk.download(omw-1.4)
!pip install -U gensim==3.8.3

stopwords = set(nltk.corpus.stopwords.words(english))
#store a list of English stop words to be used later in a variable named stopwords.
```

First we load the csv file, and select the text whose type is paragraph as dataframe.

```
import pandas as pd
import numpy as np
import re  # Import regular expression package

# Store in a pandas dataframe
data = pd.DataFrame(pd.read_csv(/content/Chegg-ESG-Report-2021 (1).csv))
df_para = data[data[type]==paragraph]
df_para.reset_index(drop=True, inplace=True)
df=pd.DataFrame(df_para, columns=[text])
```

	text
0	Over the last decade we have focused on puttin...
1	The pace of evolution within the learning ecos...
2	Weve also seen the challenges of the last few ...
3	Chegg seeks to always serve as a valued and re...
4	We know that the heart of Chegg is our incredi...
...	...
201	Corporate Giving Research Advocacy Community E...
202	(1) In 2020 and 2021 respectively, 4 and 17 of...
203	Total Board Members GRI 102 9 10 9 Board of Di...
204	Code of Business Conduct and Ethics GRI 205 2 ...
205	Please see our Policies and ESG Resources page...

206 rows × 1 columns

Then we import the re package and then use the sub function which removes parts of the string that match with the specified regular expression.

```
def remove_url(text):
  return re.sub(rhttps?:\S*,,text) # Remove url

df.text = df.text.apply(remove_url)

# Remove mentions and hashtags
def remove_mentions_and_tags(text):
     text = re.sub(r@\S*,,text)
     text = re.sub(r#\S*,,text)
     return re.sub(r[^\w ]+, "", text) # only match Unicode word characters
[a-zA-Z0-9_]

df.text = df.text.apply(remove_mentions_and_tags)

display(df.text)
```

```
0      Over the last decade we have focused on puttin...
1      The pace of evolution within the learning ecos...
2      Weve also seen the challenges of the last few ...
3      Chegg seeks to always serve as a valued and re...
4      We know that the heart of Chegg is our incredi...
                              ...
201    Corporate Giving Research Advocacy Community E...
202    1 In 2020 and 2021 respectively 4 and 17 of ou...
203    Total Board Members GRI 102 9 10 9 Board of Di...
204    Code of Business Conduct and Ethics GRI 205 2 ...
205    Please see our Policies and ESG Resources page...
Name: text, Length: 206, dtype: object
```

Several topic extraction models exist, here we tested LDA, BERTopic, NMF, and compared their results with GPT-3s.

1. Latent Dirichlet Allocation (LDA)

The basic assumption for LDA is that each of the documents can be represented by the distribution of topics which in turn can be represented by some word distribution.

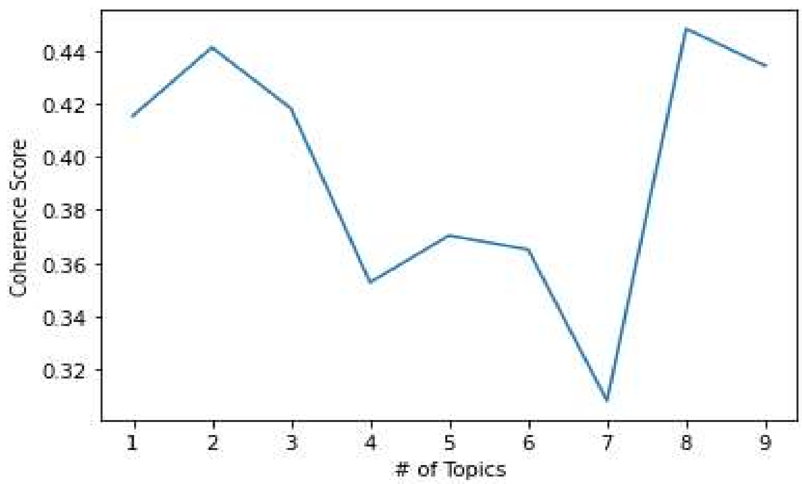
We first tokenize and lemmatize the data, transform it to a gensim dictionary and then create a variable called “bow_corpus” in which we store the Bag-of-Words (bow) transformed documents. This step is necessary because of the way the gensim package accepts inputs.

```
import gensim
def text_preprocessing(df):
    corpus=[]
    lem = WordNetLemmatizer()
    # For Lemmatization
    for text in df[text]:
        words=[w for w in nltk.tokenize.word_tokenize(text) if (w not in
stopwords)]
        # word_tokenize function tokenizes text on each word by default
        words=[lem.lemmatize(w) for w in words if len(w)>2]
        corpus.append(words)
    return corpus

# Apply this function on our data frame
corpus = text_preprocessing(df)

# Transform to gensim dictionary
dic = gensim.corpora.Dictionary(corpus)
bow_corpus = [dic.doc2bow(doc) for doc in corpus]
import pickle # Useful for storing big datasets
pickle.dump(bow_corpus, open(corpus.pkl, wb))
dic.save(dictionary.gensim)
```

After that, we use a loop to visualize the effect of different numbers of topics on the coherence score.

```
from gensim.models import CoherenceModel
import matplotlib.pyplot as plt
topics = []
score = []
for i in range(1,10,1):
     lda = gensim.models.LdaMulticore(corpus=bow_corpus, id2word=dic,
iterations=10, num_topics=i, workers = 3, passes = 10, random_state=42)
```

```
	cm = CoherenceModel(model=lda, corpus=bow_corpus, texts=corpus,
coherence=c_v)
	topics.append(i) # Append number of topics modeled
	score.append(cm.get_coherence()) # Append coherence scores to list
plt.plot(topics, score)
plt.xlabel(# of Topics)
plt.ylabel(Coherence Score)
plt.show()
```



- **Coherence score** was applied to evaluate the association of the extracted topics so that we can choose a suitable number of topics.
  C_v is one of the more widely used approaches, normally somewhere ***between 0.5 and 0.7*** would be considered as a decent score.
  (https://stackoverflow.com/questions/54762690/evaluation-of-topic-modeling-how-to-understand-a-coherence-value-c-v-of-0-4)
- **num_topics**: specify the number of topics to be extracted from the corpus.
- **workers**: specify the number of processors that will participate in the multi-processing operation.
- **passes**: controls how often we train the model on the entire corpus. Another word for passes might be “epochs”.

We can see that none of the choices can reach the decent range, here we choose num_topics = 8 since it has the highest score. But from my observation, eight topics are a bit too many and I cant see the difference between some topics very clearly.

```
lda_model = gensim.models.LdaMulticore(corpus=bow_corpus, id2word=dic,
iterations=10, num_topics = 8, workers = 3, passes=10, random_state=42)

### Exploring Common Words For Each Topic With Their Relative Words
for idx, topic in lda_model.print_topics():
    print("Topic: {} \nWords: {}".format(idx, topic ))
    print("\n")
```

```
Topic: 0
Words: 0.021*"Above" + 0.017*"Best" + 0.015*"Female" + 0.013*"Male"
+ 0.013*"Global" + 0.010*"Manager" + 0.010*"Director" +
0.009*"Staff" + 0.009*"Technical" + 0.009*"support"


Topic: 1
Words: 0.040*"employee" + 0.009*"inclusive" + 0.009*"program" +
0.008*"global" + 0.008*"diverse" + 0.008*"culture" + 0.008*"Cheggs"
+ 0.007*"development" + 0.007*"Chegg" + 0.007*"Ambassador"


Topic: 2
Words: 0.012*"Board" + 0.010*"support" + 0.010*"including" +
0.008*"Chegg" + 0.008*"People" + 0.008*"student" + 0.008*"Help" +
0.008*"board" + 0.008*"Learners" + 0.007*"ESG"


Topic: 3
Words: 0.041*"Chegg" + 0.017*"data" + 0.016*"2021" + 0.015*"tCO" +
0.014*"GRI" + 0.012*"help" + 0.012*"Study" + 0.010*"employee" +
0.010*"2022" + 0.008*"2020"


Topic: 4
Words: 0.009*"help" + 0.008*"say" + 0.008*"need" + 0.008*"new" +
0.008*"Chegg" + 0.007*"Workplaces" + 0.007*"Small" + 0.007*"Medium"
+ 0.007*"example" + 0.007*"CBD"


Topic: 5
Words: 0.016*"Global" + 0.016*"Chegg" + 0.011*"Turnover" +
0.009*"education" + 0.009*"worker" + 0.008*"employee" +
0.007*"company" + 0.007*"The" + 0.007*"digital" + 0.006*"Consumer"


Topic: 6
Words: 0.011*"employee" + 0.011*"ESG" + 0.010*"among" +
0.010*"training" + 0.009*"topic" + 0.009*"stakeholder" +
0.009*"Employees" + 0.008*"conducted" + 0.008*"experience" +
0.007*"key"


Topic: 7
Words: 0.016*"employee" + 0.014*"learner" + 0.013*"student" +
0.013*"learning" + 0.011*"education" + 0.010*"Our" + 0.009*"Chegg" +
0.008*"time" + 0.007*"global" + 0.007*"helping"
```

The well-trained LDA model can also be used for classification of unknown text:

```
from operator import itemgetter

unseen_document = The feedback we received reinforced our belief that Cheggs
mission and values are critical to our business success and are deeply
```

```
integrated into our culture and processes.

def para_preprocess(text):
    result = []
    lem = WordNetLemmatizer() # For Lemmatization
    text = re.sub(rhttps?:\S*,,text) # Remove url
    text = re.sub(r@\S*,,text) # Remove mentions
    text = re.sub(r#\S*,,text) #Remove hashtags
    text = re.sub(r[^\w ]+, "", text) # only match Unicode word characters
[a-zA-Z0-9_]
    words=[w for w in nltk.tokenize.word_tokenize(text) if (w not in stopwords)]
    # word_tokenize function tokenizes text on each word by default
    words=[lem.lemmatize(w) for w in words if len(w)>2]
    result.append(words)

    return result

def topic_classification(txt):
  bow_corpus=para_preprocess(unseen_document)
  bow_vector = dic.doc2bow(bow_corpus[0]) # transform the corpus to doc2bow
  print(lda_model.get_document_topics(bow_vector))
  index = max(lda_model.get_document_topics(bow_vector),key=itemgetter(1))[0] #
find the most similair topic
  print("Topic{}: {}".format(index, lda_model.print_topic(index, 10))) # print
the top words in this topic
  return lda_model.print_topic(index, 10)

res=topic_classification(unseen_document)
```

```
[(0, 0.94094193), (1, 0.012136078)]
Topic0: 0.021*"Above" + 0.017*"Best" + 0.015*"Female" + 0.013*"Male"
+ 0.013*"Global" + 0.010*"Manager" + 0.010*"Director" +
0.009*"Staff" + 0.009*"Technical" + 0.009*"support"
```

2. BERTopic

BERTopic is a topic modeling python library that combines transformer embeddings and clustering model algorithms to identify topics in NLP. Because the embedding vectors usually have very high dimensions, dimension reduction techniques are used to reduce the dimensionalities.
The default algorithm for dimension reduction is UMAP (Uniform Manifold Approximation & Projection). Compared with other dimension reduction techniques such as PCA (Principle Component Analysis), UMAP maintains the datas local and global structure when reducing the dimensionality, which is important for representing the semantics of the text data.

requirement

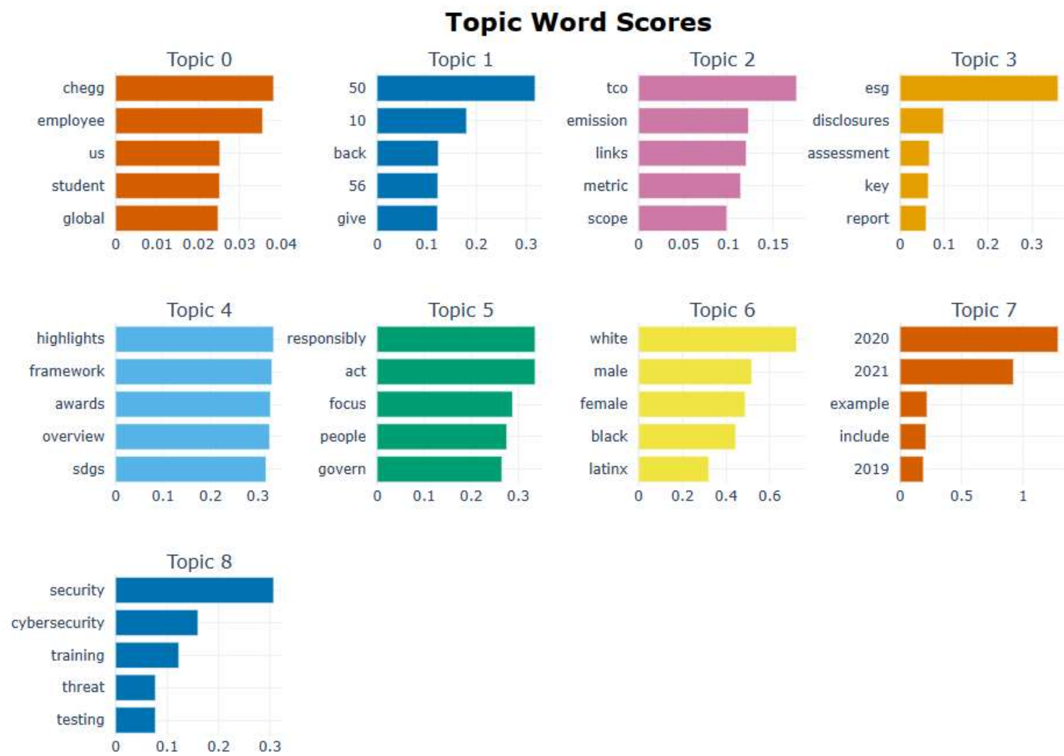
```
!pip install bertopic
!pip install --upgrade joblib==1.1.0
```

BERTopic will automatically determine the number of topics based on the training data.

```
# Topic model
from bertopic import BERTopic
# Dimension reduction
from umap import UMAP

lem = WordNetLemmatizer()
df_input = data[text].apply(lambda x:  .join([w for w in x.split() if w.lower()
not in stopwords]))
# Lemmatization
df_input_lem = df_input.apply(lambda x:  .join([lem.lemmatize(w) for w in
x.split() if w not in stopwords]))

# Initiate UMAP
umap_model = UMAP(n_neighbors=15,
                  n_components=5,
                  min_dist=0.0,
                  metric=cosine,
                  random_state=100)
# Initiate BERTopic
topic_model = BERTopic(umap_model=umap_model, language="english",
calculate_probabilities=True)
# Run BERTopic model
topics, probabilities = topic_model.fit_transform(df_input_lem)
# Get the list of topics
topic_model.get_topic_info()
```

	Topic	Count	Name
0	-1	27	-1_say_coursework_understand_chegg
1	0	259	0_chegg_employee_us_student
2	1	75	1_50_10_back_56
3	2	36	2_tco_emission_links_metric
4	3	34	3_esg_disclosures_assessment_key
5	4	30	4_highlights_framework_awards_overview
6	5	24	5_responsibly_act_focus_people
7	6	16	6_white_male_female_black
8	7	14	7_2020_2021_example_include
9	8	13	8_security_cybersecurity_training_threat

```python
# Visualize top topic keywords
topic_model.visualize_barchart(top_n_topics=10)
```



Like LDA, BERTopic can also classify the topic of unknown text.

```python
new_review = "The feedback we received reinforced our belief that Cheggs
mission and values are critical to our business success and are deeply
integrated into our culture and processes."
# Find topics
num_of_topics = 3
similar_topics, similarity = topic_model.find_topics(new_review,
top_n=num_of_topics);
# Print results
print(fThe top {num_of_topics} similar topics are {similar_topics}, and the
similarities are {np.round(similarity,2)})
# Print the top keywords for the top similar topics
for i in range(num_of_topics):
  print(fThe top keywords for topic {similar_topics[i]} are:)
  print(topic_model.get_topic(similar_topics[i]))
```

```
The top 3 similar topics are [4, 3, 0], and the similarities are
[0.28 0.27 0.27]
The top keywords for topic 4 are:
[(highlights, 0.33332467418214173), (framework,
0.32994210545593106), (awards, 0.32667194273220235), (overview,
0.3252125540833023), (sdgs, 0.3174695866613478), (pillars,
0.3154073929828161), (materiality, 0.31458557196024445), (oversight,
0.30906292061057355), (disclosures, 0.2988858404806048), (capital,
```

0.021435091302897225)]
The top keywords for topic 3 are:
[(esg, 0.360531237433342), (disclosures, 0.09900062445779519), (assessment, 0.06670067078813456), (key, 0.06451340029147333), (report, 0.059467890023657786), (see, 0.05812406712027032), (materiality, 0.052100440789361956), (students, 0.05201799042720433), (stakeholders, 0.05196884144122686), (formal, 0.05196884144122686)]
The top keywords for topic 0 are:
[(chegg, 0.03823097894112883), (employee, 0.03559785559731583), (us, 0.02523146371985384), (student, 0.02519395201719064), (global, 0.024797022667829113), (education, 0.02307621561858618), (data, 0.022555989019899154), (support, 0.022239347898639714), (learner, 0.021740528431069814), (learning, 0.019873417726543715)]

3. Non-Negative Matrix Factorization (NMF)

Non-negative matrix factorization is also a supervised learning technique which performs clustering as well as dimensionality reduction. It can be used in combination with TF-IDF scheme to perform topic modeling. In this section, we will see how Python can be used to perform non-negative matrix factorization for topic modeling.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vect = TfidfVectorizer(max_df=0.8, min_df=2, stop_words=english)
doc_term_matrix = tfidf_vect.fit_transform(df_input_lem.values.astype(U))
from sklearn.decomposition import NMF

nmf = NMF(n_components=8, random_state=42)
nmf.fit(doc_term_matrix)
for i,topic in enumerate(nmf.components_):
    print(fTop 10 words for topic #{i}:)
    print([tfidf_vect.get_feature_names_out()[i] for i in
topic.argsort()[-10:]])
    print(\n)
```

Top 10 words for topic #0:
[time, improve, cost, chegg, academic, learning, support, education, learner, student]

Top 10 words for topic #1:
[help, operate, sustainably, focus, people, learners, responsibly, act, govern, effectively]

Top 10 words for topic #2:
[studying, use, grade, need, better, coursework, help, understand, chegg, say]

Top 10 words for topic #3:
[result, engagement, invite, response, respondent, approximately, reflect, results, conducted, survey]

Top 10 words for topic #4:
[12, 2020, 30, 11, 20, 31, 2021, excludes, ux, data]

Top 10 words for topic #5:
[ambassador, 1154, inclusive, workplace, cybersecurity, program, diversity, global, training, employee]

Top 10 words for topic #6:
[stakeholder, including, topic, oversight, sustainability, board, governance, committee, director, esg]

Top 10 words for topic #7:
[invested, im, approach, marketing, responsible, security, tc, privacy, goal, policy]

For all paragraphs in the data, NMF can classify their topics.

```
topic_values = nmf.transform(doc_term_matrix)
df_para[Topic] = topic_values.argmax(axis=1)
df_para.head(20)
```

	Unnamed: 0	page_number	type	text	Topic
0	3	2	paragraph	Over the last decade we have focused on puttin...	0
1	4	2	paragraph	The pace of evolution within the learning ecos...	0
2	6	2	paragraph	Weve also seen the challenges of the last few ...	0
3	7	2	paragraph	Chegg seeks to always serve as a valued and re...	0
4	9	3	paragraph	We know that the heart of Chegg is our incredi...	5
5	10	3	paragraph	In 2021, Chegg collectively donated 1,400,000 ...	0
6	11	3	paragraph	The challenges of the last few years have had ...	6
7	12	3	paragraph	We are a mission driven company with an enormo...	0
8	13	3	paragraph	Sincerely,	0
9	14	3	paragraph	Dan Rosensweig CEO, President, and Co Chairper...	2
10	15	4	paragraph	When it comes to our most valuable resource, o...	5
11	17	4	paragraph	At Chegg, we take our position within the educ...	0
12	19	4	paragraph	The viability and health of societies will soo...	0
13	20	4	paragraph	Without a major boost to digital skills, as a ...	0
14	38	6	paragraph	Chegg is a mission driven company . We strive ...	0
15	39	6	paragraph	We aim to support and accelerate the path stud...	0
16	40	6	paragraph	This sentiment is weaved into everything we do...	6
17	42	6	paragraph	We are committed to making a difference on the...	6
18	46	7	paragraph	Formal responsibilities for the implementation...	6
19	47	7	paragraph	Cheggs Governance and Sustainability committee...	6

We could define a class for GPT-3